Mannan Abdul

21801066

CS 202

HW 2

Q1)

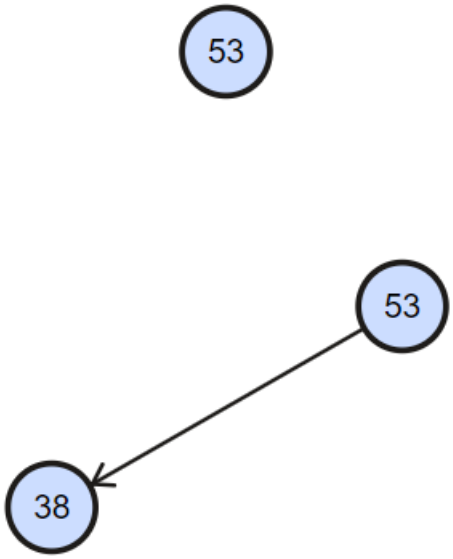
Prefix expression: - x x A B - + C D E / F + G H

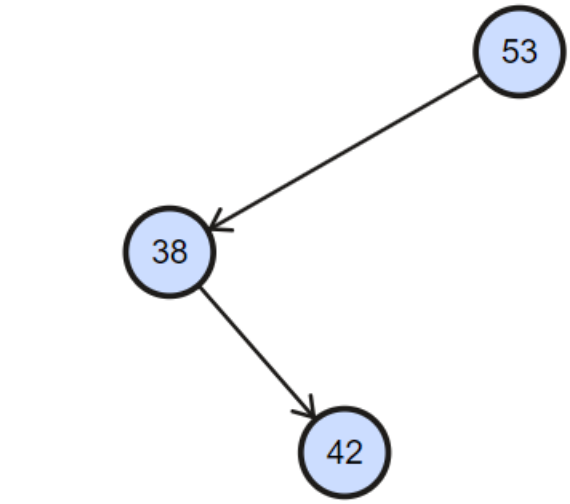
Infix expression: A x B x ((C + D) – E)) – ((F / (G + H))

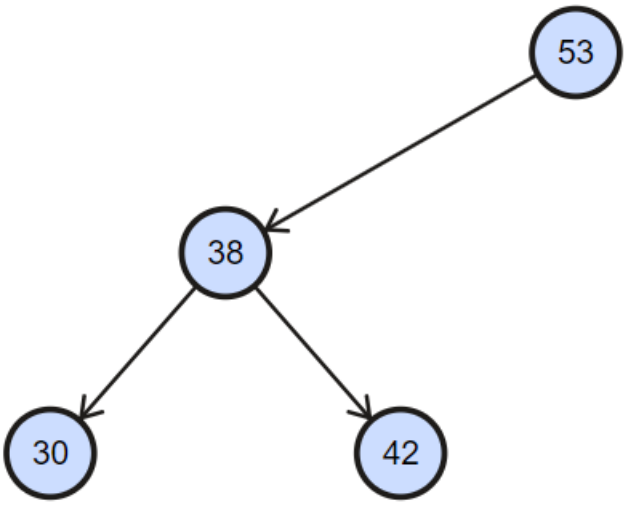
Postfix expression: A B x C D + E – x F G H + / -

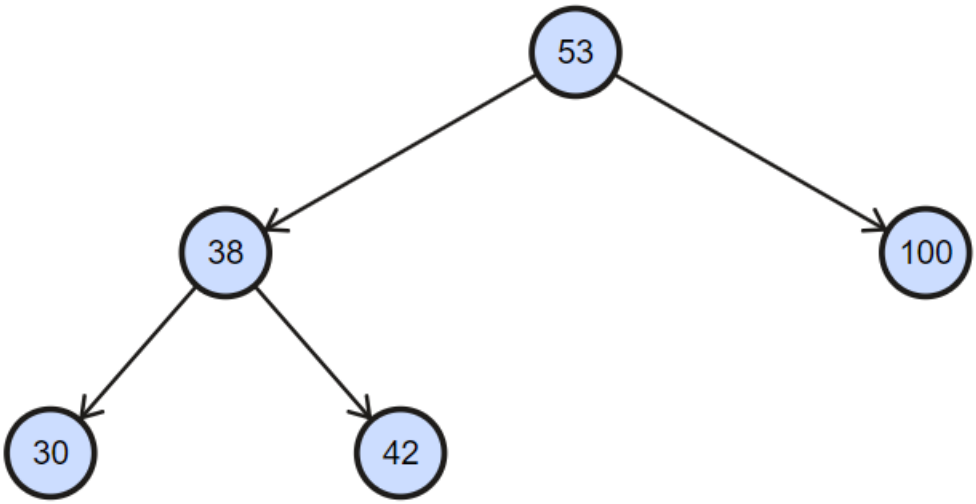
Q2)

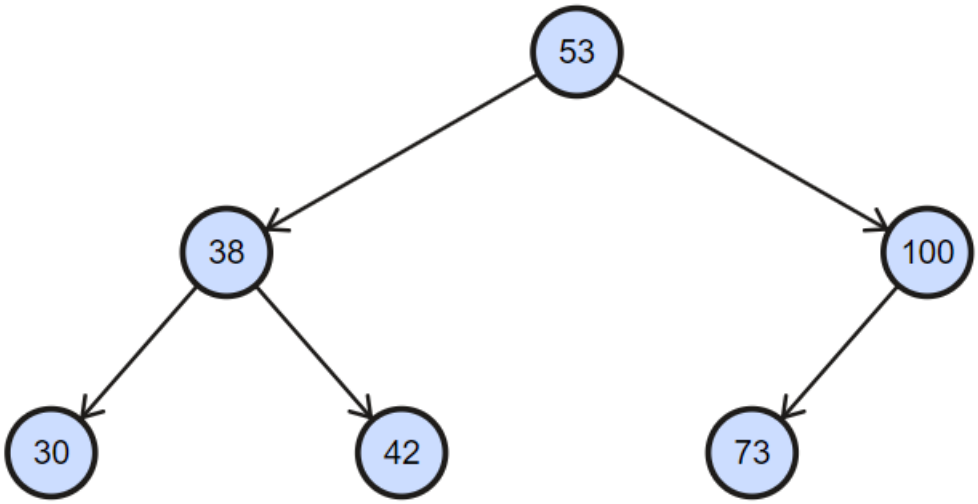
Insertions for 53, 38, 42, 30, 100, 73, 111, 94, 33, 86, 63, 23, 83, 101

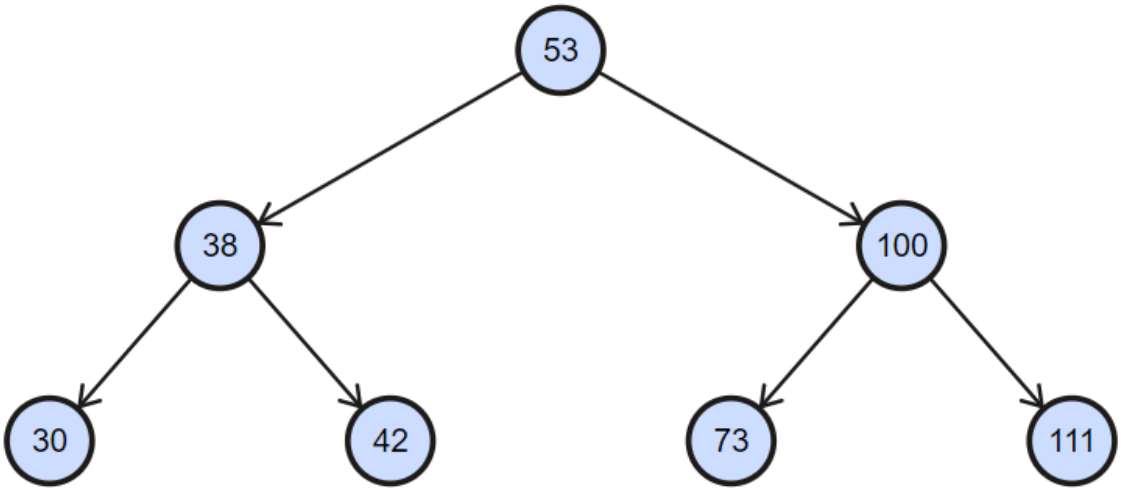


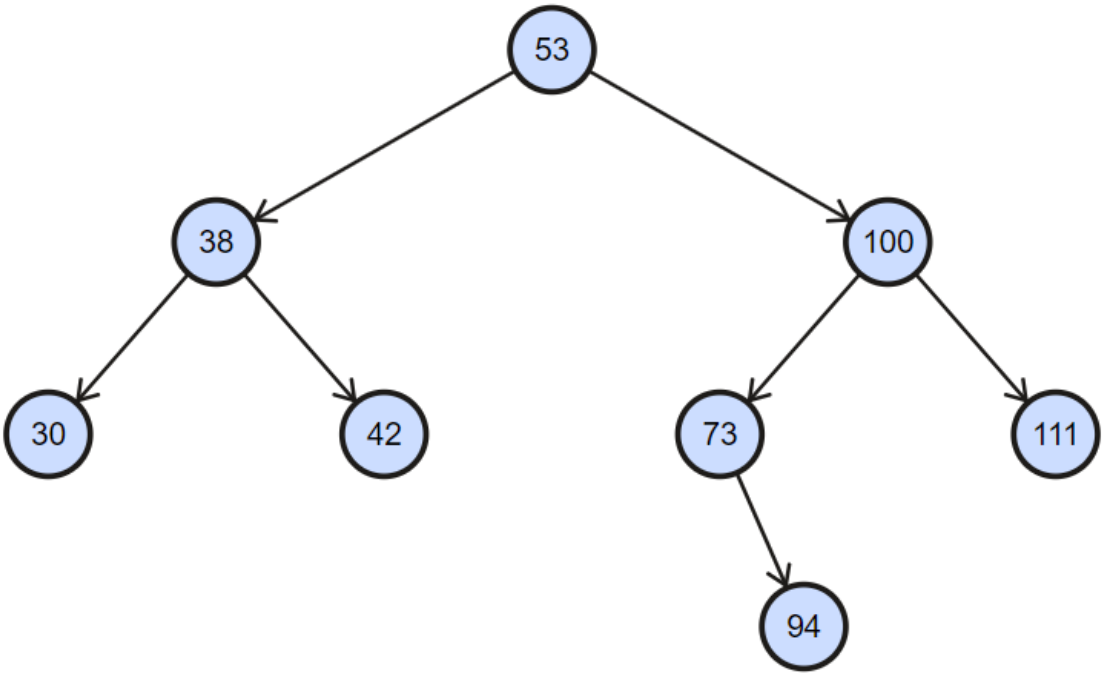

53
38
42

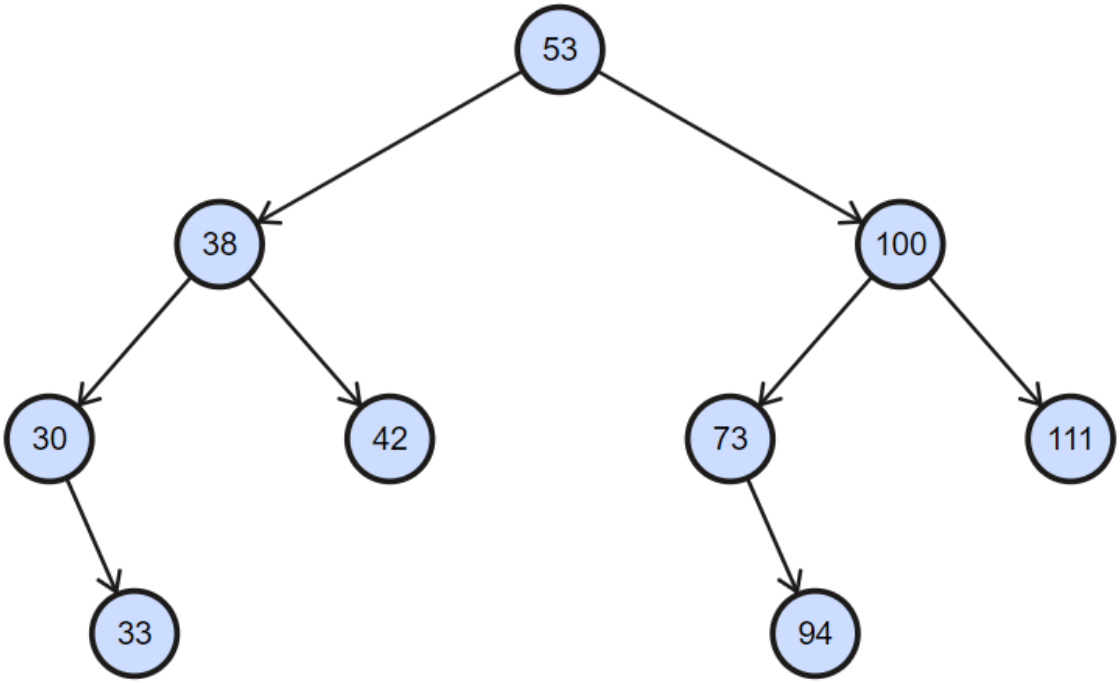

53
38
30
42

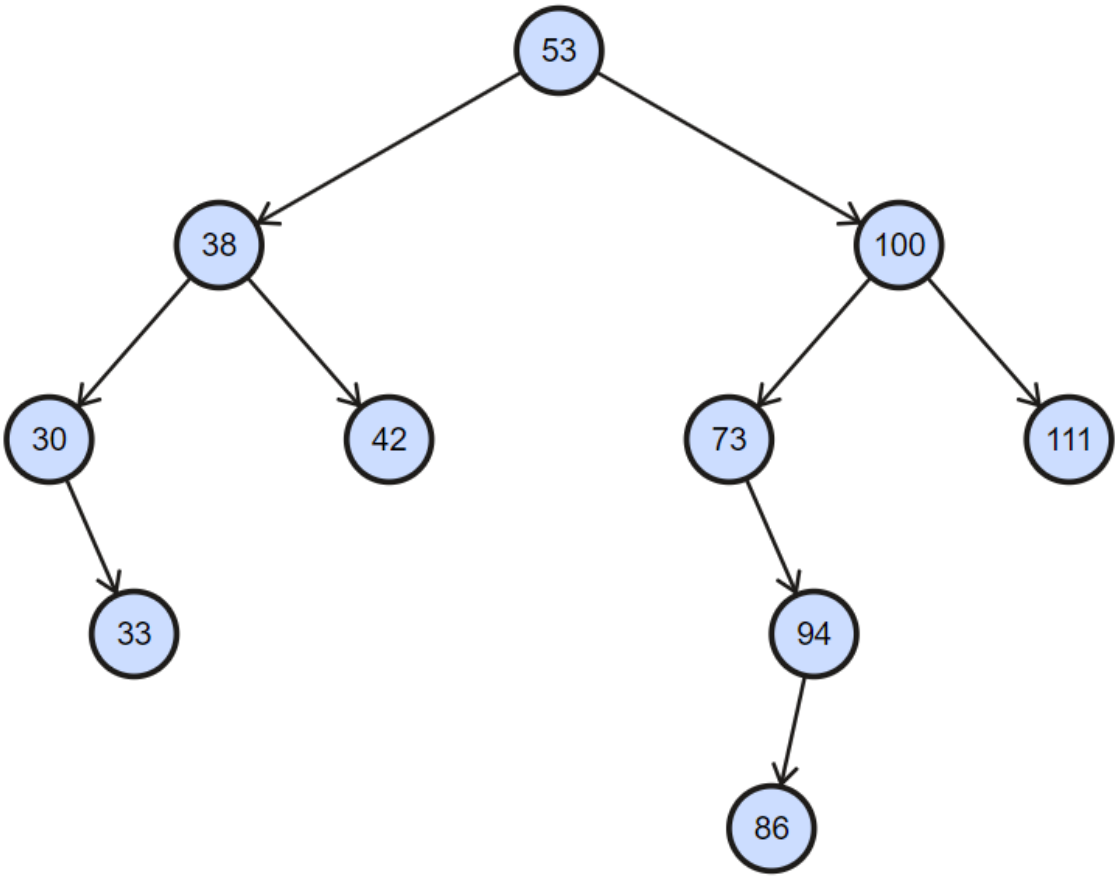

53
38
100
30
42

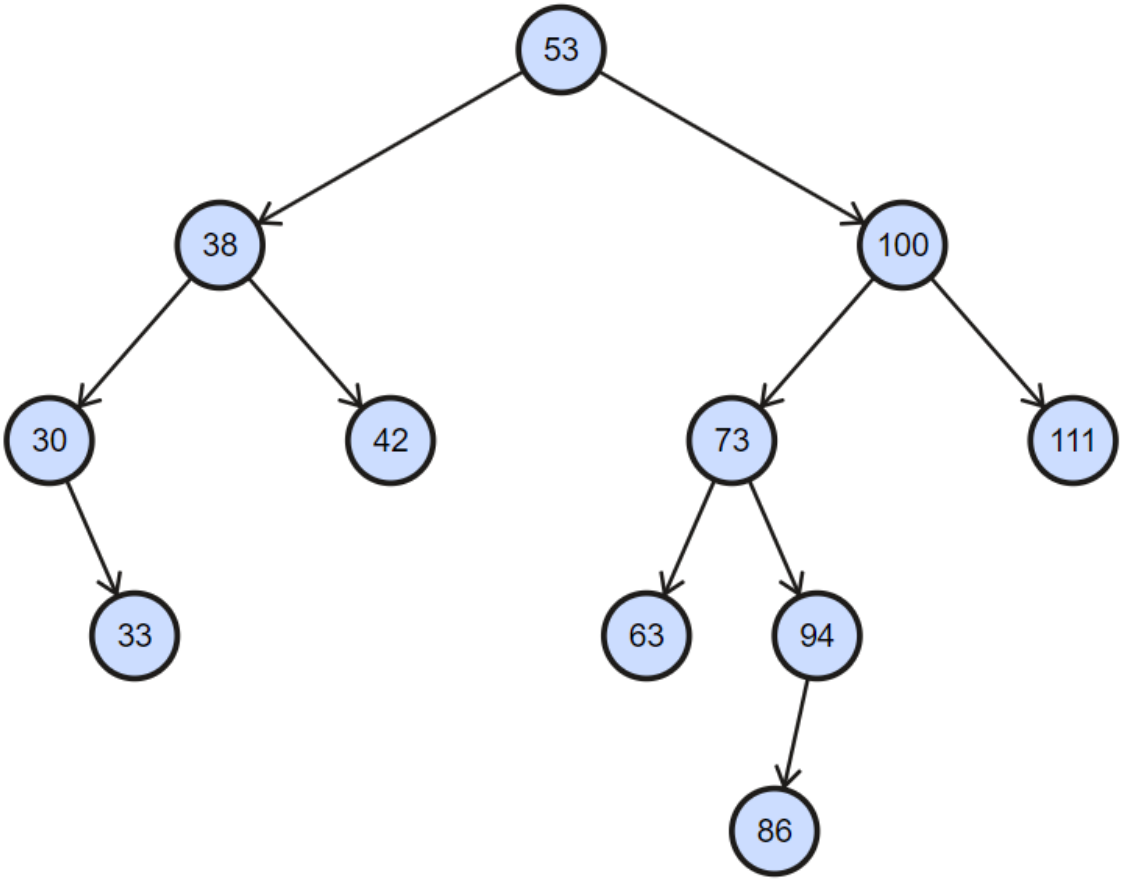

53
38
100
30
42
73

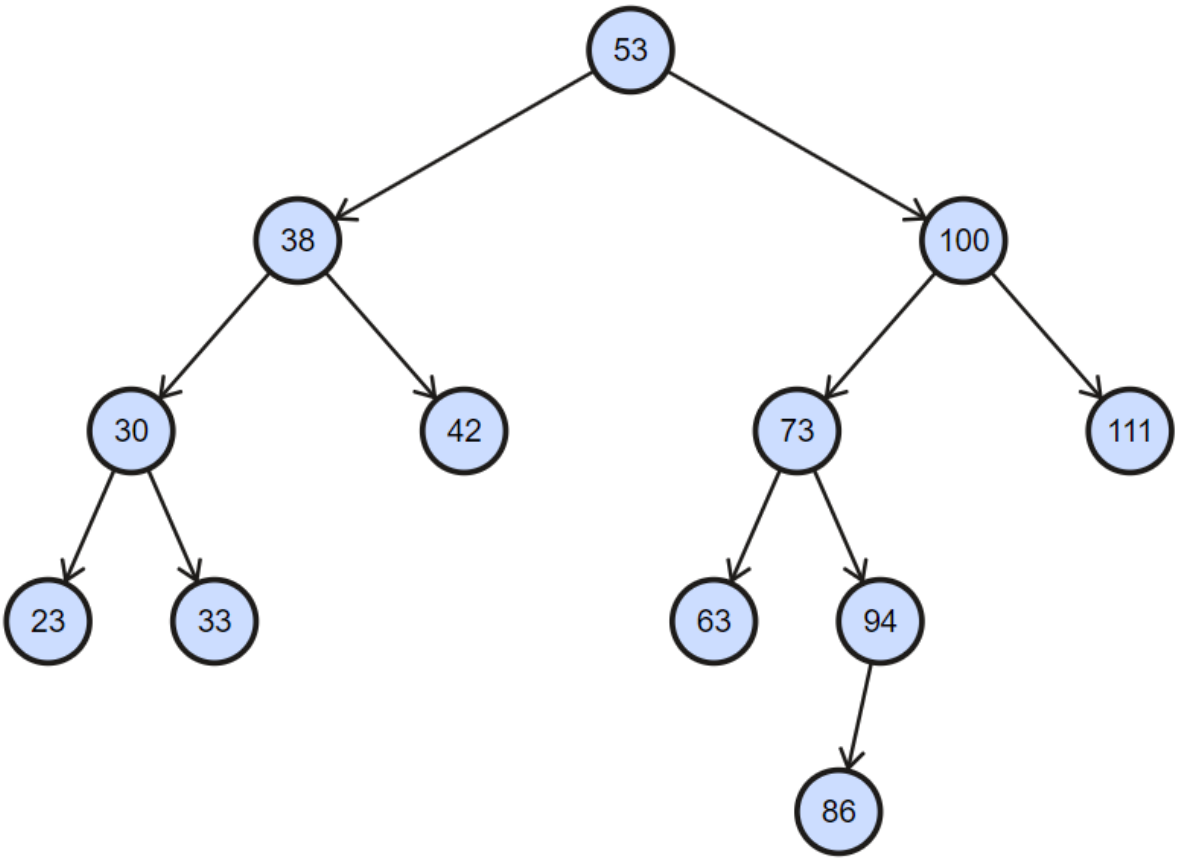

53
38
100
30
42
73
111

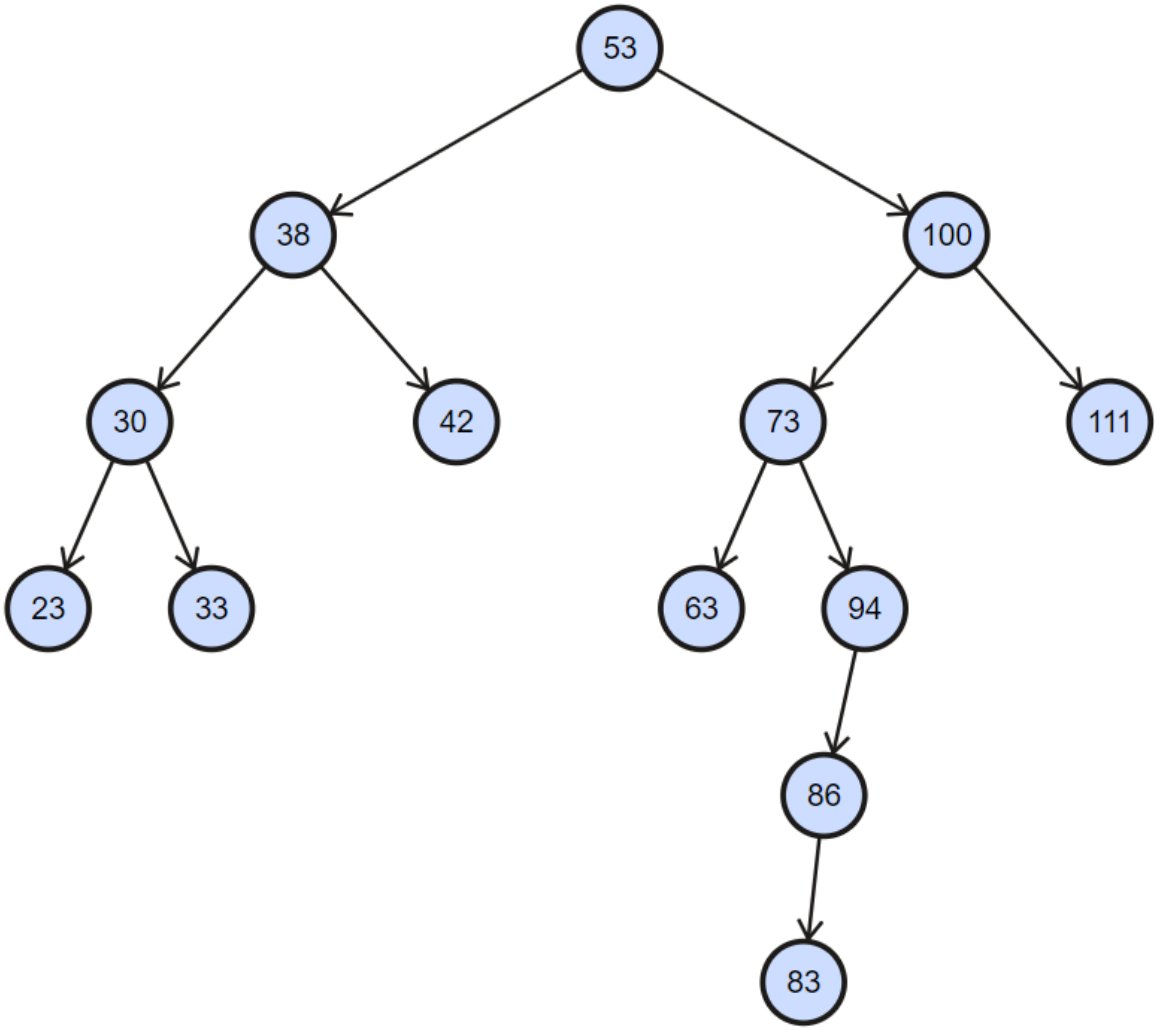

53
38
100
30
42
73
111
94

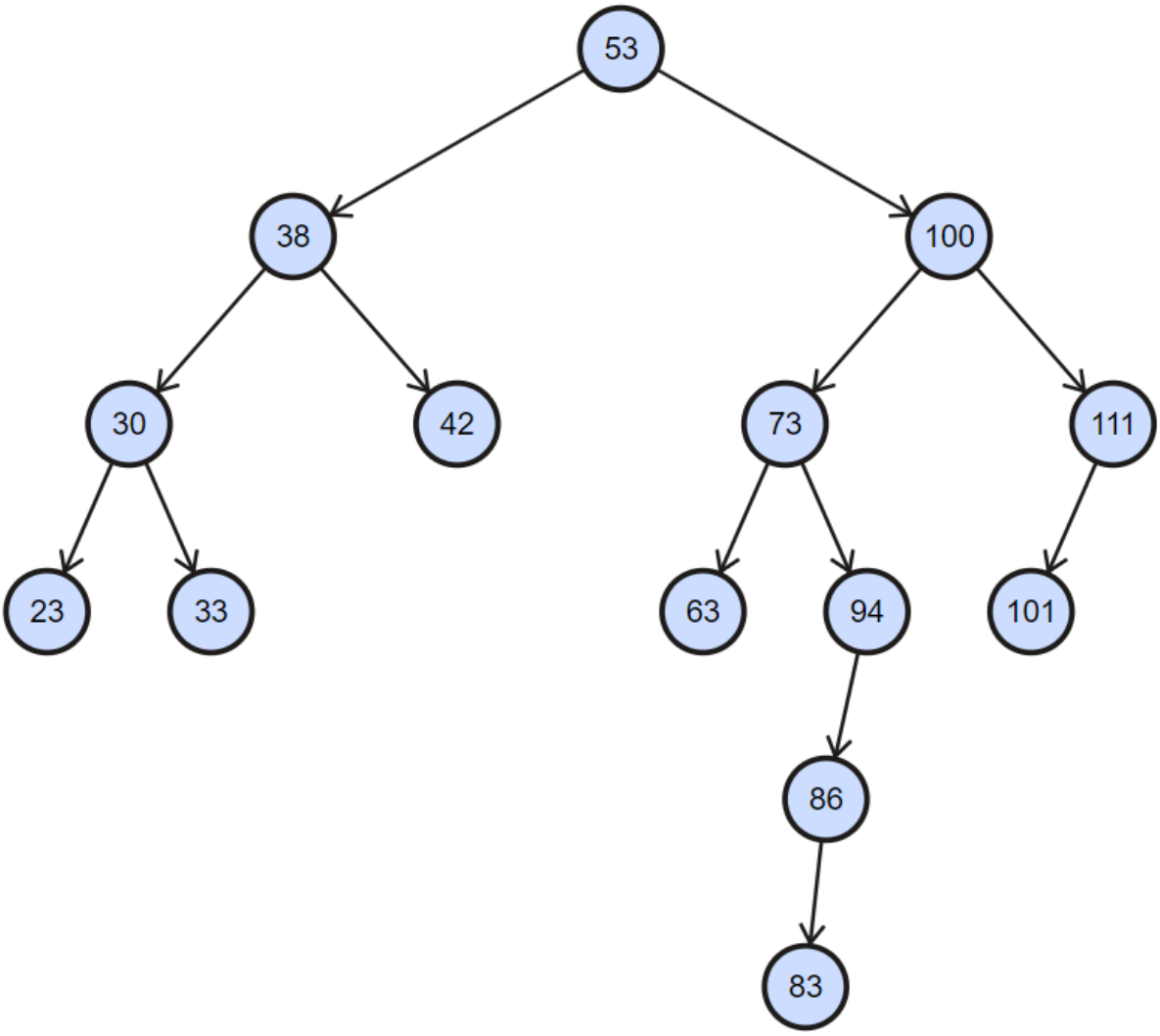

53
38
100
30
42
73
111
33
94


53
38
100
30
42
73
111
33
94
86

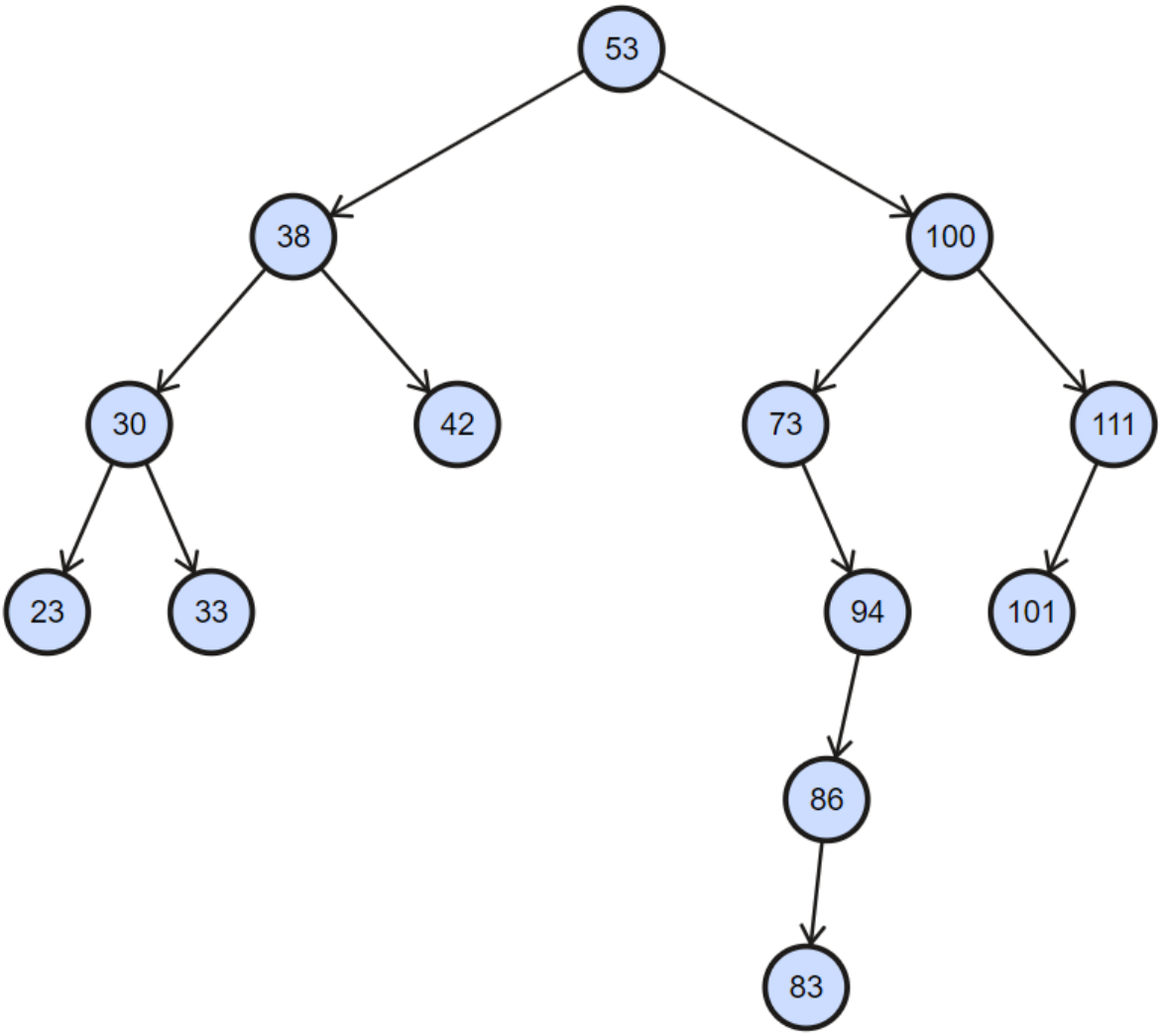

53
38
100
30
42
73
111
33
63
94
86

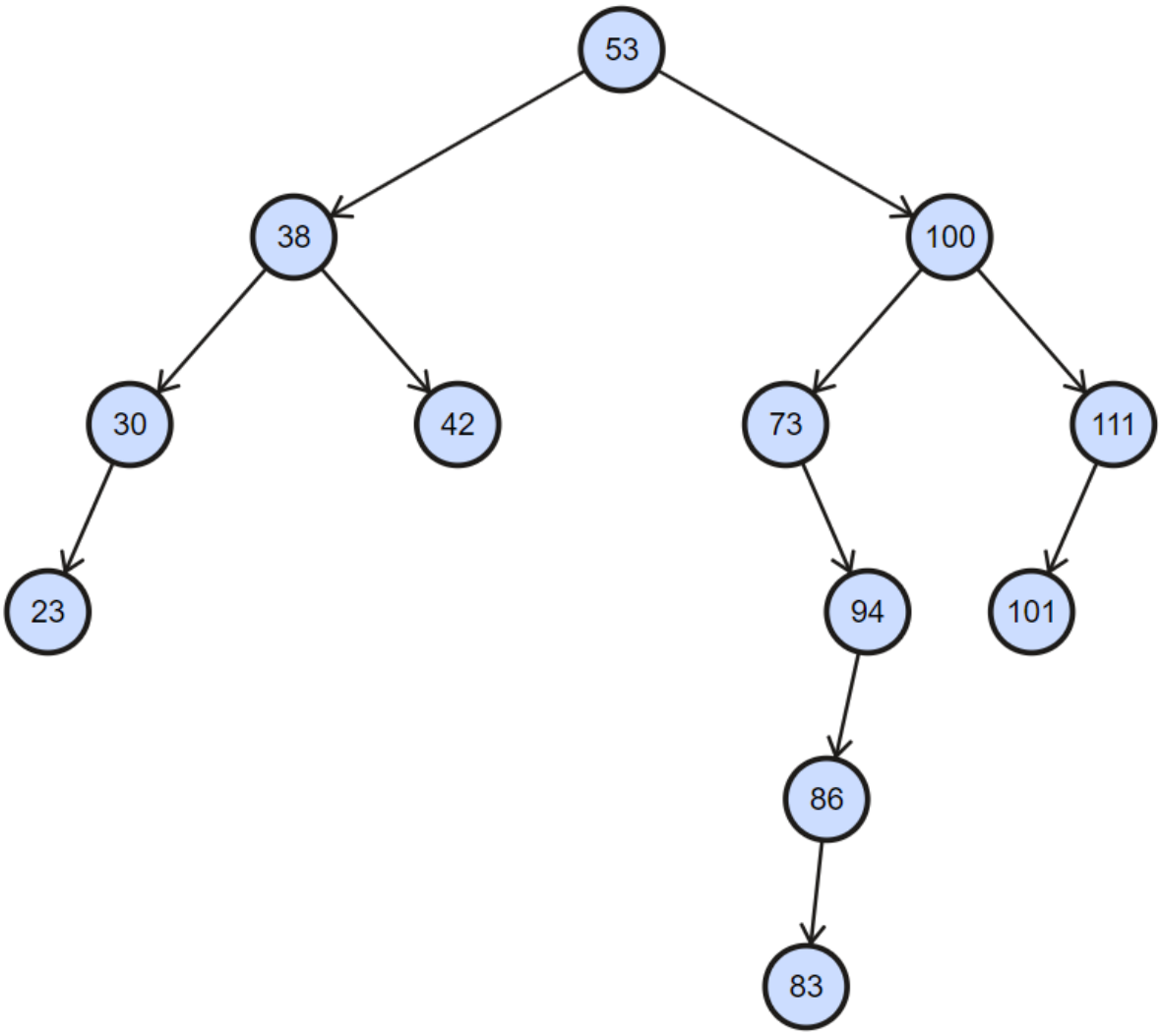

53
38
100
30
42
73
111
23
33
63
94
86

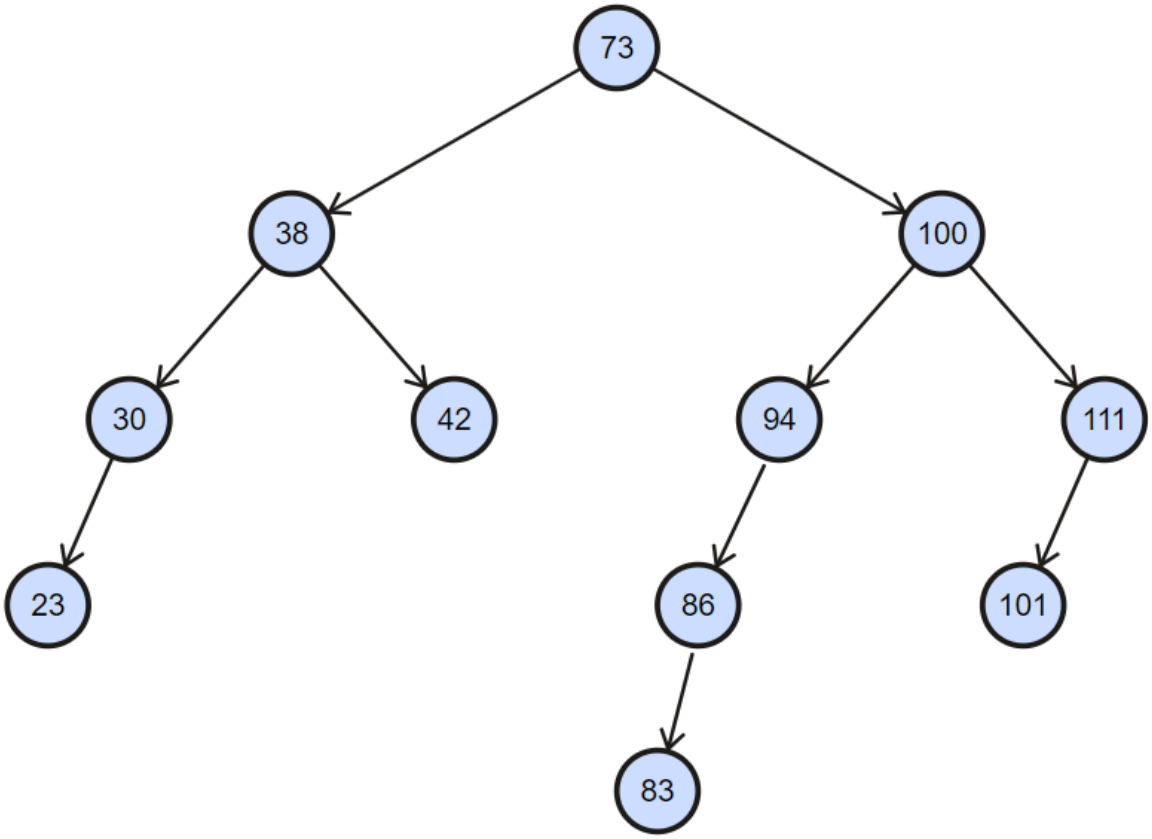

53
38
100
30
42
73
111
23
33
63
94
86
83


53
38
100
30
42
73
111
23
33
63
94
101
86
83

After deletions: 63, 33, 53




53
38
100
30
42
73
111
23
94
101
86
83



Q4)

The addNgram function is a method that is meant for the insertion of a single object into the binary tree, as such its worst case running time complexity would be **O(n)**. Even though we implement the function recursively, in the worst case, all n nodes of our BST would be on one side and that means to check for pre-existing ngrams ot to add a new ngram at the the end, we would have to traverse all n nodes which provides us with our worst case complexity. Similarly, for the << operator function outputs each and every ngram, therefore, it too has to traverse over all the elements in the BST. If we assume a similar worst case scenario where all n nodes of the BST are all on the same side, then again we would have to traverse the entire height of the BST and get the worst case running time of **O(n).**

Therefore, the worst case running time for both the addNgram and << operator function are both **O(n)**